

AI Secure Phish

MS. T. Renuka¹, Velakkayala Abhinav², Surapuraju Mokshit Varma³, Pillarikuppam Yashwanth Reddy⁴, Sadar Wilson Raj⁵

¹Professor, ²³⁴⁵Students, Computer Science and Engg., Hyderabad Institute of Technology and Management. Gowdavelly Village, Medchal Mandal, Medchal-Malkajgiri District, Hyderabad, Telangana, India.

ABSTRACT - Phishing is among the gravest problems facing cybersecurity today. Fraudsters publish counterfeit websites designed to harvest private data, including login credentials, banking information and personal details. Most current defenses use blacklists or hand-written rules, and both perform poorly against phishing pages that are newly created or keep changing. To address this weakness, AI Secure Phish applies machine learning to judge a website from its URL and associated characteristics, such as URL length, domain composition, whether HTTPS is used, suspicious words and special characters. The classifiers compared in this work are Random Forest, XGBoost, LightGBM, SVM, KNN and Naïve Bayes. Beyond simple classification, the system offers side-by-side model comparison, risk evaluation, a record of previous scans, graphical threat analysis and explanations for each result. Users can reach it through a web application that gives detailed website analysis, or through a browser extension that raises an alert during normal browsing. Development relied on React.js, FastAPI, Python and Scikit-learn.

Keywords - Phishing Detection, Machine Learning, Cybersecurity, URL Analysis, Random Forest, XGBoost, Browser Extension, Risk Assessment.

1. INTRODUCTION

Phishing has become a serious hazard in cyberspace as the internet has expanded. Attackers use forged web pages and deceptive links to trick people into disclosing confidential information [1]. With online services now part of everyday life, identifying these fraudulent sites early matters more than ever.

Several defenses are already in use: blacklists, rule-based engines, protections built into web browsers, and expert systems [1]. A blacklist recognizes only those sites that someone has previously reported. A rule-based engine, meanwhile, checks pages against a fixed set of conditions. Newly created phishing sites are therefore often absent from every blacklist, and rigid rules lose accuracy as attackers repeatedly alter their methods.

AI Secure Phish responds to this shortcoming by using

machine learning to detect phishing [2], [3]. The features it considers include URL length, domain structure, HTTPS usage, special symbols and suspicious keywords. Several machine learning models are employed, and the results they produce are merged with a risk analysis.

2. PROBLEM STATEMENT

Phishing is a leading cybersecurity concern. Attackers create fake websites and malicious links that closely resemble authentic ones so as to collect usernames, passwords, bank details and other personal information. Growing dependence on online services leaves more people open to such attacks, and the consequences include monetary loss and identity theft. Detection today depends mainly on blacklists and rule-based methods. Both handle already-known phishing sites well. They fail, however, when a site is new and absent from every blacklist, or when it does not fit any existing rule. Genuine and fraudulent sites are also visually very similar, so users struggle to spot a phishing page before they enter sensitive information.

What is required is an intelligent, automated system that inspects a website's characteristics, flags probable phishing sites and explains the associated risk clearly. AI Secure Phish meets this need by combining machine learning classification with risk assessment, analytical reporting and detection inside the browser.

3. OBJECTIVES

This work aims to develop a system for detecting and preventing phishing, using AI and machine learning to recognize phishing websites and improve web security.

The proposed system is intended to achieve the following:

- To examine phishing attacks and the harm they cause to individuals and organizations.
- To extract and analyze URL features that are useful for spotting phishing.
- To develop models and measure how effectively they detect phishing websites.
- To give users a prediction of whether a website is phishing or legitimate.

- To keep a log of completed scans and present insights on a dashboard.
- To build the web application using React.js, FastAPI, Python, PostgreSQL and Scikit-learn.
- To make users more aware of cybersecurity threats.

4. PROPOSED METHODOLOGY

The Phishing Detection System uses a centralized machine learning approach. A URL can be typed into the web application, or the browser extension can send the URL of the page being visited. On the backend, the system pulls out features including length, domain, HTTPS, keywords and symbols. Machine learning modules then assign the URL one of three labels: legitimate, suspicious or phishing. Once a prediction exists, the user is shown the risk information and the scan is recorded in a PostgreSQL database.

The overall process is summarized below:

Website URL → Feature Extraction → Machine Learning Model → Prediction → Risk Evaluation → Result / Browser Warning → Scan History

Fig. 1 presents the architecture, which comprises the user interface, backend server, feature extraction module, ML models, database and browser extension. The main advantage of the system is that these parts form one integrated platform, bringing together multiple ML models, risk analysis, scan history, graphical analytics and live browser detection. This gives it more functionality than typical phishing classifiers.

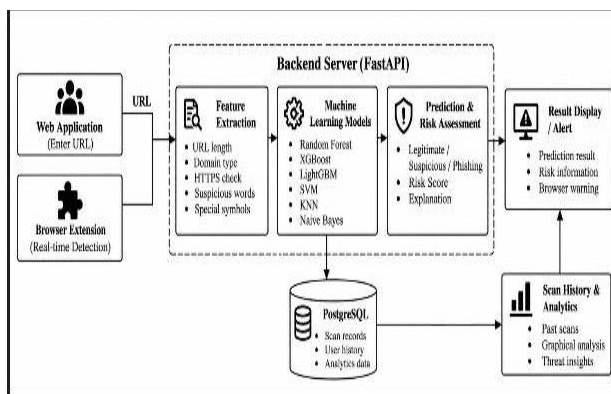


Figure 1: Centralized architecture of the proposed system

```
BEGIN AISecurePhish
INPUT url, userId, sourceType // sourceType: "WEB_APP" or "BROWSER_EXTENSION"
// Step 1: Validate input URL
IF url IS NULL OR url == "" THEN
    OUTPUT "Invalid URL. Please enter a valid website address."
    TERMINATE
ENDIF
// Step 2: Extract features from the URL
features = EXTRACT_URL_FEATURES(url)
IF features IS EMPTY THEN
    OUTPUT "Feature extraction failed."
    TERMINATE
ENDIF
// Step 3: Load trained machine learning model
model = LOAD_MODEL()
IF model IS NULL THEN
    OUTPUT "Model loading failed."
    TERMINATE
ENDIF
// Step 4: Predict the website class
prediction, confidence = PREDICT(model, features) // prediction: Legitimate/Suspicious/Phishing
IF prediction == "Phishing" THEN
    result = "Phishing"
    riskMessage = "High Risk - This is a phishing website."
ELSE IF prediction == "Suspicious" THEN
    result = "Suspicious"
    riskMessage = "This website may be unsafe."
ELSE
    result = "Legitimate"
    riskMessage = "This website appears to be safe."
ENDIF
// Step 5: Store scan details in database
storeStatus = STORE_SCAN_HISTORY(userId, url, features, result, confidence, sourceType)
IF storeStatus == "SUCCESS" THEN
    OUTPUT "Scan details stored successfully."
ELSE
    OUTPUT "Failed to store scan details."
ENDIF
// Step 6: Generate response for user
response = GENERATE_RESPONSE(result, confidence, riskMessage)
OUTPUT response
// Step 7: If real-time scanning (browser extension), show alert
IF sourceType == "BROWSER_EXTENSION" THEN
    SHOW_BROWSER_ALERT(result, riskMessage)
ENDIF
// Step 8: Return scan history if requested
IF USER_REQUESTS_HISTORY(userId) == TRUE THEN
    historyData = GET_SCAN_HISTORY(userId)
    DISPLAY_HISTORY(historyData)
ENDIF
END AISecurePhish
```

Figure 2: Overall pseudocode of the proposed phishing detection system

The main stages of the proposed system are shown in Figure 2: checking the website URL, extracting its features, applying the trained ML model, producing a phishing prediction, storing the scan details, and presenting the result through the web application or browser extension.

4.1 URL Feature Analysis

The system inspects a number of characteristics of the submitted URL for indications of phishing. The features identified include URL length, domain type and structure, whether HTTPS is present, suspicious keywords and unusual symbols [7], [8]. The ML models receive these features and use them for classification, deciding whether the site is legitimate, suspicious or phishing and giving the user the related risk information.

4.2 Phishing Detection Workflow

1. A URL is entered in the web app, or the browser extension passes along the URL of the page currently open. The URL is sent on to the FastAPI backend, where it is analyzed.
2. Important URL and website features are pulled out.
3. Trained machine learning models receive the extracted features.
4. Each website is classified by the models as legitimate, suspicious or phishing.
5. A prediction and the associated risk information

are produced.

6. The user sees the result in the web application, or as a warning in the browser.

7. Details of the scan are saved in PostgreSQL so they can be used later.

The workflow is expressed as follows:

Website URL → Feature Extraction → ML Models → Prediction → Risk Assessment → Results / Browser Warning → Scan History

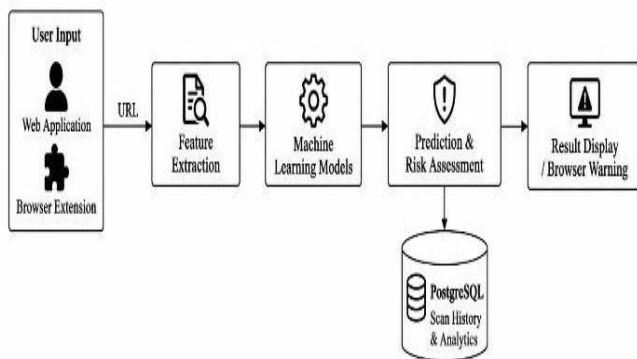


Figure 3: Workflow of the proposed system

Figure 3 depicts the workflow used to analyze a website URL with machine learning. It runs from feature extraction through prediction and risk assessment, ending with the result shown to the user.

5. IMPLEMENTATION ENVIRONMENT

AI Secure Phish follows a web-based architecture that combines machine learning, backend processing, database management and detection inside the browser. React.js provides the web interface and FastAPI powers the backend services. Feature processing and model development are done in Python with Scikit-learn. PostgreSQL holds scan records, prediction results and related data, and a browser extension checks for phishing in real time as pages are visited.

Component	Technology / Platform	Purpose/Description
Web Application	React.js	Interface for submitting URLs for scanning
Backend	FastAPI	Receives requests and runs the prediction process
ML Development	Python, Scikit-learn	Feature engineering and model training

ML Models	RF, XGBoost, LightGBM, SVM, KNN, NB	Classification of websites
Database	PostgreSQL	Keeps scan history and prediction results
Browser Extension	HTML, CSS, JavaScript	Real-time URL checking while browsing

The system is built as a web-based phishing detector on React.js, FastAPI, Python, Scikit-learn and PostgreSQL. Users enter URLs in the web app and receive phishing predictions in return. Each URL is passed to the FastAPI backend, which extracts the required features and feeds them to the trained classifiers. The models investigated for this task include Random Forest, XGBoost, LightGBM, SVM, KNN and Naive Bayes [2], [4]. A browser extension can additionally be developed so URLs are checked live during browsing. PostgreSQL stores scan records and prediction results. The components communicate via the centralized backend, which delivers phishing predictions, risk information, scan history and graphical analytics.

6. SYSTEM MODELING

6.1 Overview

AI Secure Phish has a centralized design that connects URL analysis, machine learning prediction, risk scoring and data storage. Users may submit a URL in the web application, or the browser extension may hand over the URL while they browse. The backend handles the URL, extracts its features and forwards them to the trained models. The prediction is supplemented with risk information before it goes back to the user. Scan details are also kept in PostgreSQL, which supports scan history and graphical analytics.

The complete flow, from submitting a URL to showing the phishing prediction, is handled by the system. Since the architecture is centralized, the web app and the browser extension rely on the same backend and machine learning services.

6.2 Use Case Model

The use case model describes how people interact with AI Secure Phish. There are three main actors: the User, the Browser Extension and the Administrator. A User can submit a URL, run a scan, and look at prediction results, risk scores, scan history and graphical analytics. The Browser Extension reads the address of the website that is open, starts a scan and warns the

user about unsafe pages. An Administrator can manage the machine learning models, examine scan results and access analytics on how the system performs.

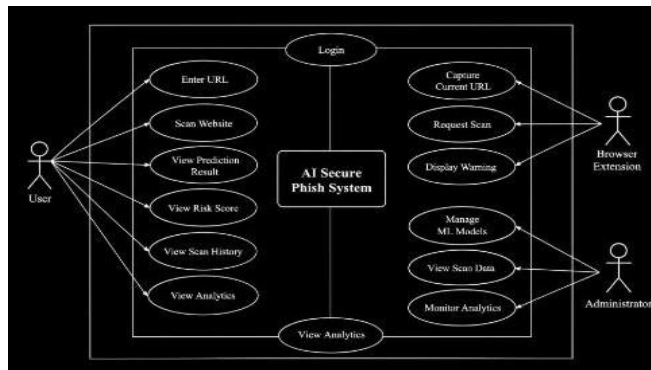


Figure 4: Use case model of the AI Secure Phish system

6.3 System Components and Data Flow

The sequence of steps in the system is as follows:

1. To begin, the user supplies the URL to be checked through the web application or the browser extension.
2. The React.js web application accepts the URL and readies it for scanning.
3. The FastAPI backend receives that URL and scans it for malicious content.
4. Features relevant to identifying malicious content are then extracted from the URL.
5. A set of classifiers (machine learning models) predicts whether the URL is malicious.
6. The prediction result is assembled, containing the predicted probability and information about the URL examined.
7. This result is stored in the PostgreSQL database to allow later analysis.
8. The user then receives the outcome through the web application or the browser extension, and the extension can show a phishing warning if the URL turns out to be malicious.
9. Scan results also feed the graphical statistics on malicious URLs.

The diagram below shows the architecture of the system.

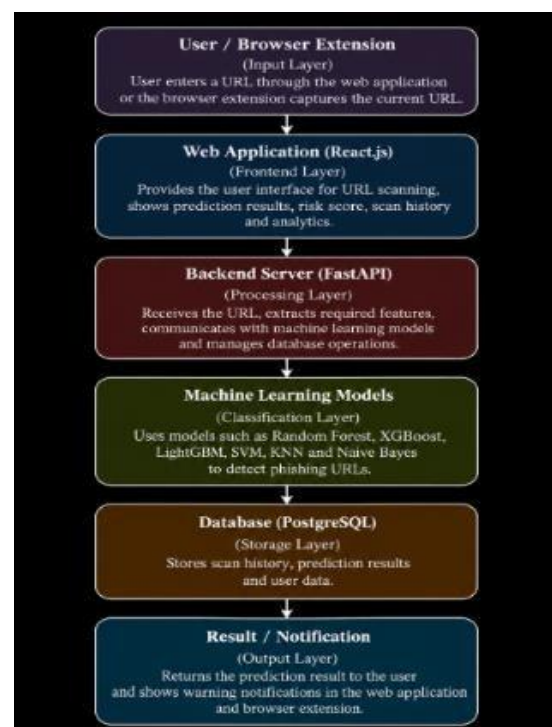


Figure 5: Data flow of the AI Secure Phish system

7. RESULTS AND DISCUSSION

AI Secure Phish takes a unified approach to identifying and evaluating phishing websites. The web application provides phishing predictions, risk assessment and scan details, and the extension identifies phishing sites while users are online.

With this approach, several machine learning techniques can be compared for effectiveness, along with prediction accuracy, the dependability of risk evaluation and the value of scan history and graphical analysis. Measures such as accuracy, precision, recall and detection time will be reported only once practical evaluation has been completed; this paper presents no hypothetical data.

AI Secure Phish consists of interconnected modules that cooperate to detect phishing websites. Data moves from the user or browser extension through the web application, backend and machine learning models to the database, and results are sent back to the user as predictions and warnings.

8. IMPLEMENTATION STAGES

Stage	Activity
Requirement Analysis	Analyze phishing attacks, URL features, user needs and detection requirements.

System Design	Design the web application, FastAPI backend, ML models, database and browser extension.
Feature Extraction	Derive URL length, domain structure, HTTPS usage, keywords and special symbols.
ML Integration	Train and evaluate the Random Forest, XGBoost, LightGBM, SVM, KNN and Naive Bayes models.
Database Integration	Set up PostgreSQL to store scan history and prediction results.
Browser Integration	Build browser extension support for real-time URL scanning and warnings.
Testing	Check prediction performance, risk assessment, data storage and browser detection.
Deployment	Get the complete system ready for practical use once testing succeeds.

9. FUTURE SCOPE

Future work could add more advanced machine learning models, a refined risk assessment model and analysis of further website and URL features. Other possible improvements include a stronger browser extension, multilingual interfaces, clearer prediction explanations and links to real-time threat intelligence data. Adaptive models that learn to recognize new phishing variants could also raise detection performance over time. Such extensions should be attempted only after the existing ML models, centralized backend, web application and browser-based detection are complete.

10. CONCLUSION

AI Secure Phish is a centralized system for identifying phishing websites with artificial intelligence and machine learning. By evaluating several URL parameters, it determines whether a website is genuine, suspicious or phishing.

Several machine learning models are combined in the tool with risk measures, scan history, graphical analytics and detection carried out within the browser. A single web application, supported by a centralized FastAPI backend, brings all these functions together.

Because everything is integrated into one application, AI Secure Phish offers an effective and user-friendly response to phishing. Testing in

practice will show how well the models and other components perform.

10.1 SYSTEM INTERFACE AND IMPLEMENTED SCREENS



Figure 6: Main user interface of the AI Secure Phish system

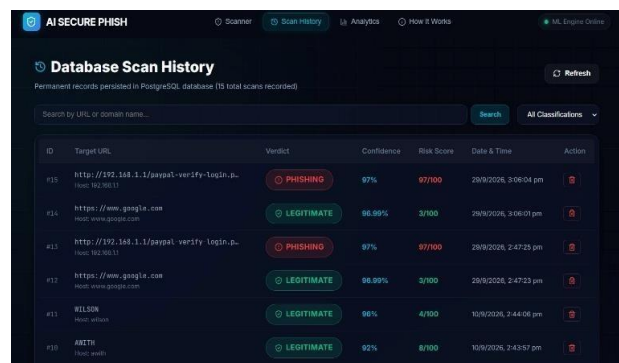


Figure 7: Database scan history and phishing detection results



Figure 8: Threat intelligence and security analytics dashboard

11. REFERENCES

- 1) K. L. Chiew, K. S. C. Yong, and C. L. Tan, "A survey of phishing attacks: Their types, vectors and technical approaches," Expert Systems with Applications, vol. 106, pp. 1–20, 2018.

- 2) O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.
- 3) A. Prasad and S. Chandra, "PhiUSIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," *Computers & Security*, vol. 136, 103545, 2024.
- 4) S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit*, 2007.
- 5) M. Mohammad and L. McCluskey, "Phishing Websites," *UCI Machine Learning Repository*, 2012.
- 6) N. Abdelhamid, "Website Phishing," *UCI Machine Learning Repository*, 2014.
- 7) A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using Lexical Features for Malicious URL Detection A Machine Learning Approach," *arXiv:1910.06277*, 2019.
- 8) "URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis," *2021 12th International Conference on Information and Communication Systems (ICICS)*, IEEE, 2021.